MECHANISM FOR EFFICIENT WEAROUT COUNTERS IN DESTRUCTIVE READOUT MEMORY

BACKGROUND

This application is a continuation of prior U.S. Serial No. 09/966,499 filed September 27, 2001.

1. Field of the Invention

This disclosure relates to destructive readout memories, more particularly to mechanisms to track and level usage in these memories.

2. Background of the Invention

Polymer ferroelectric memories generally comprise a polymer material having ferroelectric properties arranged adjacent to some type of control structures. The control structures may be as simple as metal electrode lines. The control structures manipulate the ferroelectric state of the polymer material. In memory applications, the ferroelectric state having one characteristic is determined to be a data 'one,' and a different characteristic is determined to be a data 'zero.'

These memories are relatively inexpensive on a cost/bit basis, and reasonable simple to manufacture. However, the ferroelectric properties of the polymer material degrade after extensive use. It would be helpful to track access of the memory array to manage usage and wear out of blocks within the array.

It would also be useful to perform the tracking with minimal impact on processing efficiency, as well as requiring a minimal amount of storage.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention may be best understood by reading the disclosure with reference to the drawings, wherein:

Figure 1 shows a block diagram of one embodiment of a memory block on a ferroelectric memory chip.

Figure 2 shows a flowchart of one embodiment of a memory access cycle, in accordance with the invention.

Figure 3 shows a flowchart of one embodiment of a read cycle, in accordance with the invention.

Figure 4 shows a flowchart of one embodiment of a write cycle, in accordance with the invention.

Figure 5 shows a flowchart of one embodiment of a method to provide a wear out counter using a minimal number of bits, in accordance with the invention.

DETAILED DESCRIPTION OF THE EMBODIMENTS

Memory devices, such as disks or disk caches, are typically laid out in blocks of 512 bytes. Each block of data has metadata associated with it, which may include an error correction code (ECC) that allows checking and correction of errors in that data block. Typically, the ECC is a pre-existing data set, even if the memory has never been accessed, even if the pre-existing data is all zeros. If the memory block is being used as cache memory, there will often be flags and other information may also reside in the metadata section allowing the system to make cache policy decisions. This data may include usage information for replacement of data, etc. This section may also include a wear out counter that tracks the usage history of the block of memory.

In Figure 1, a memory chip 10 has at least one block of memory that resides upon it. Each block of memory has a sector of memory cells 12 and a metadata section 14. The metadata section may include the ECC, flags for cache operations, etc., as well as a wear-out counter 16. The wear out counter 16 will more than likely

include just the data storage for the wear out counter, as the logic that actually reads and increments, decrements or resets the counter will usually lie in the memory controller 18. Techniques for minimizing the amount of storage required for the memory controller will be discussed further with reference to Figure 5.

5   The memory controller may not be a separately dedicated controller, but the system processor or another system controller that also handles memory control. However, for purposes of this discussion, this will be referred to as a memory controller, even though it may have many other functions. Similarly, while the memory controller is shown as residing adjacent the blocks of memory in the

10   embodiment of Figure 1, the memory controller 18 may reside separately from the memory, such as on a host or system board. Alternatively, it may even comprise software control on a processor.

In one embodiment the memory blocks and memory chip are comprised of polymer ferroelectric memory. This type of memory, like other types of memory

15   cells that have a finite lifetime can also degrade over time or 'wear out.' The counter used to track the number of times a particular block has been accessed is referred to here as a 'wear out counter.' A wear out counter may be used for any block or other division of a memory that can degrade over time and/or with extensive use, such as repeated access.

20   A flowchart of one embodiment of a method for tracking memory access using a wear out counter is shown in Figure 2. The block of memory participates in a memory access cycle at 20. The memory access cycle may be a read cycle or a write cycle as will be discussed with reference to Figures 3 and 4. The counter is incremented to reflect the new access cycle at 22. After the counter is updated the

25   error correction code, if used, is updated at 24 to reflect the changes to the memory

data caused by the memory access cycle and to the metadata caused by the incremented counter.

At 26, the counter is checked against a predetermined threshold of a number of access cycles. The predetermined threshold is a number set depending upon the characteristics of the memory. A memory block that has participated in a memory access cycle more times than that set by the threshold may be in risk of wearing out or failing. At this point the memory controller or software will "remap" this sector of memory and transfer its contents to a new memory location that has not exceeded its wearout threshold. If the counter has not exceeded the threshold at 26, the memory block can continue to participate in memory access cycles with no rectification.

In one embodiment, the memory access cycle may be a read cycle. This is shown in more detail in Figure 3. The memory access cycle of 20 in Figure 2 is shown in Figure 3 as one embodiment of a read cycle in Figure 3. At 30, a sector of memory is read. The sector may be a block, more than one block, or part of a block. At 32, the block data is corrected using the appropriate ECC for the data. If ECC is used to correct the data, the sector of memory will be an entire block, as ECC can only be used on an entire block. Correction of the block data with the ECC is optional, as some memories may be 'good' enough to not require use of ECC.

In this example, the sector will be a block of data, as shown in Figure 1. Typically, this block is 512 Kbytes, but no limitation on the size of the blocks is intended by this example. For that block, the ECC located in the metadata section is used to correct the data, as discussed above. No correction may be necessary, but the correction is performed as needed.

As mentioned previously, the memory block may comprise at least a portion of a polymer ferroelectric memory array. Polymer ferroelectric memories generally

comprise a layer of polymer material having ferroelectric properties sandwiched between two layers of electrode lines, where the electrode lines may be arranged as word and bit lines. Polymer ferroelectric memories do not require any transistors for each cell, relying instead upon alterations of the ferroelectric state of a local area of

5   the polymer material to store ones and zeros. One state is determined to be a one, and the opposite polarization state of the region of the polymer between two particular electrodes is determined to be a zero.

These memories have several advantages over conventional memory products available currently. With the lack of transistors, they are relatively simple to

10   manufacture and therefore can be manufactured inexpensively and with higher yields, as there are no circuit components for each cell that may fail. However, when the cells of this type of memory array are accessed for a read, they undergo a 'destructive' read. Generally, the process of sensing the polarization state of the cell returns the polarization state of the cell to that state associated with a zero. Therefore, during a

15   read cycle such as that shown in Figure 3, the data must be written back to the memory cells after the read. It must be noted that the example of polymer ferroelectric memories is given only one type of destructive read memories. Application of the invention is not limited to polymer ferroelectric memories, but may be used in any memory having a destructive read cycle. This includes the

20   embodiments discussed with regard to Figure 3.

In the example of Figure 3, the data is written back at 34. In the context of the process shown in Figure 2, this would occur after the ECC and other metadata are updated to reflect the incrementing of the counter at 34. As destructive read memories already require a write back of the metadata, updating the counter does not

contribute to any significant increase in overhead, making the process of using a wear out counter transparent with regard to the performance of the memory.

In a write process, an example of which is shown in Figure 4, the approach is similar. However, in order to write to a destructive read memory, such as the polymer

5 ferroelectric memory, it is generally desirable to first 'zero out' the memory, bringing all of the cells to a known polarization state. The destructive nature of read cycles in these memories allows this to be accomplished by performing a read and then discarding the data.

In Figure 4, the memory access cycle 20 of Figure 2 comprises reading the

10 sector of memory at 40, as well as reading the counter at 42. Again, since the reading process has to be performed anyway, there is no significant contribution to any overhead for the wear out counter.

Referring back to Figure 2, after the memory access cycle 20 is accomplished, the counter is incremented. After the ECC, if used, and other metadata are updated at

15 24, the write of new data shown at 44 in Figure 4, would occur. The write back of the data would include the updated ECC, if used, and the incremented counter value.

As discussed above, the incrementing and writing of the counter does not contribute to any significant overhead with regard to the time and processing overhead of the memory. It is also helpful if the counter does not take up much space

20 in the metadata section of the memory block as well. However, the number of accesses after which the memory block may degrade is on the order of 10^{12}. This is referred to as the usage threshold. In order to count that many cycles, the counter would require 40 bits. If the memory chip has 1 Gigabyte of data on it, that would be 2 million blocks, each requiring 40 bits of a wear out counter, or 80 Megabits of data

25 just for the counters.

Figure 5 shows one embodiment of a method to increment a counter using minimal space while still providing a reasonably accurate indication of wear out. Note that incrementing the counter, as used here, may include decrementing a counter, such as a countdown counter, as well. The process is based upon a scaling threshold that is determined by the number of bits the system designer desires to use for the wear out counter and the usage threshold. The number of bits will also depend upon the desired accuracy. For example, assume the designer wants to determine when a block has been accessed 10^{12} times, the usage threshold, but has some leeway in accuracy. The wear out counter can indicate wear out in the range of 5×10^{11} to 5×10^{12}.

As an example, assume the usage threshold is 10^{12} and the system designer desires a wear out counter using only 4 bits. A 4-bit counter can count up to 16. Therefore, the scaling threshold is the highest number counted by the number of bits divided by the usage level being tracked, in this case $16/10^{12}$, or 1.6×10^{-11}. This scaling threshold is used as will be discussed with reference to Figure 5.

In Figure 5, a random number is generated at 50. Random, as used here, may include a true random number from a random number generator, a pseudo-random number, a number determined by algorithm, etc. A processor somewhere in the system will probably perform this task, whether it is the memory controller, a central processing unit or other processor in the system is left up to the system designer. The random number (between 0 and 1) is generated as part of the process of incrementing the counter, as shown in Figure 2. The random number is then compared to the scaling threshold. In the example discussed above, the scaling threshold is 1.6×10^{-11}.

If the random number has a predetermined relationship with the scaling threshold, the wear out counter is incremented at 52. Typically this predetermined

relationship will be if the random number is less than the scaling threshold. If the random number is not greater than the scaling threshold, the current value of the wear out counter is written back during one of the memory access cycles as discussed in Figures 3 and 4. Note that this process is referred to as incrementing the wear out counter, even if the wear out counter is not actually incremented. In either case, the wear out counter is written back at 56. In this manner, the counter will be incremented on average every $10^{12}/16$ times. In one view of this process, when the counter is incremented, it is as if the first threshold is multiplied by the second threshold to determine what the actual threshold is. When the counter indicates that it has reached 16, the number of accesses for that block is approximately 10^{12}, or whatever usage threshold is determined. The usage threshold may also be referred to as a criterion to determine if the counter needs to be incremented.

Several different factors can affect this process, depending upon the specific application. The counter cannot be too small for a particular application, or probability of it prematurely indicating wear out becomes too high. The number of bits allocated for the random number generation must be appropriate for the threshold being used. Other types of techniques can be used to increment the counter, other than that shown in Figure 5.

Referring back to Figure 2, this is just one possible implementation for incrementing the counter at 22. An alternative embodiment involves one counter that counts up to the number of bits/usage threshold, such as $16/10^{12}$. When it hits that number, it just increments the counter for whichever particular block is currently being accessed. This may in some cases average out across the device and provide a fairly accurate assessment of when the blocks are reaching their usage threshold.

Alternative embodiments could include the use of bell curves, probability theory and other statistical predictive methods.

However, tracking the individual block usage has other advantages. Unlike the example above where the wear out counter for a block is incremented in a

5   statistical fashion, rather than for a particular block, tracking the individual blocks allows usage to be leveled across the device. For example, for some reason a particular set of blocks or even an individual block may be accessed more frequently than others. Tracking actual individual usage allows the system to determine which blocks have not been used very much and direct memory traffic to them to level out

10 the usage across the device. The nature and details of these types of processes are beyond the scope of this disclosure, but may be enabled by the use of a wear out counter as discussed above.

The actual implementation of a counter and the tracking may be simple enough to be implemented in hardware using counters and registers within a

15 controller or other logic device. Alternatively, it may be implemented in processor firmware, where only the data for the wear out counter is actually stored in hardware, being stored in the metadata section of the memory block. These choices are left up to the system designer.

Thus, although there has been described to this point a particular embodiment

20 for a method and apparatus for a memory having wear out counters to track usage, it is not intended that such specific references be considered as limitations upon the scope of this invention except in-so-far as set forth in the following claims.